Congestion Control in Mobile Networks

K.R. Subramanian T.A. Dahlberg
Department of Computer Science, The University of N. Carolina at Charlotte
Charlotte, NC 28223, USA,{krs,tdahlber}@uncc.edu

Abstract

Current research is focused on adaptive resource management protocols to offer quality of service guarantees within mobile networks. The usual approach to protocol evaluation is statistical analysis of simulation results. This is insufficient for understanding complex behaviors of distributed, adaptive algorithms. We propose adaptive admission control algorithms to manage mobile network congestion caused by failures, and data visualization for algorithm analysis. Examples illustrate the dramatic impact of even simple visualizations on enabling the protocol designer to clearly understand real-time algorithm behavior under highly variable conditions.

1 Introduction

The rapid adoption of mobile communications technology in both business and consumer applications as well as the convergence of voice and data transmission poses great challenges for extending Quality of Service (QoS) guarantees to the mobile terminal. Normal operating mode for a wireless access network is characterized by bursts of demand from mobile users attempting to gain access to wireless links with varying signal quality. Network congestion can result from hardware or software failures of network components and also from increased user demand due to rush-hour traffic or highway accidents. In addition, the availability and quality of a wireless signal is affected by terrain, foliage, rain, and other environmental occurrences.

Over-allocation of resources to meet varying demand is not solely a cost issue, but is constrained by limited frequency spectrum. As a result, current research on mobile networks is focusing on the development of adaptive techniques that are not only dynamically sensitive to current conditions, but also permit scalable resource allocation policies that are designed to meet desired performance objectives.

Adaptive protocols used in mobile networks must react to a variety of changing conditions, and hence need to keep track of network activity using a number of real-time metrics to specify, for example, sampling rates and threshold values. An inherent tradeoff exists in developing an adaptive protocol that can differentiate between "real problems which must be addressed" and "burst activity that should be ignored". Also, since protocol behavior must be monitored at each network access point (or cell), protocol analysis requires interpretation of huge amounts of data which vary on a spatial and temporal basis. This points to the need for data visualization tools.

The earliest applications of data visualization algorithms were in the fields such as biomedical imaging (CT, MRI, Ultrasound) and fluid flow analysis. In recent years, visualization has been applied to more abstract types of data, such as telephone databases [4] and multivariate data from various applications. However, application of visualization to communication networks has not approached its full potential. Work on modeling performance in a telecommunication network is reported in [2]. The emphasis in this work is on visualizing backbone network topology, with some interactive query support for understanding network activity.

2 Network Architecture

The portion of the mobile network that is of interest here is the cell-site. A cell refers to a basestation (BS) transmission tower and the geographical region surrounding it, within which a mobile terminal can reliably communicate with a BS. The resources of interest here are the wireless channels, and each BS has a fixed number of channels available. When a mobile initiates or accepts a call it makes a new-call request to the closest BS. When it moves from one cell to another, a handover-call request is made to the BS in the new cell.

A Channel Allocation (CA) policy is used by each BS to respond to new-call and handover-call requests. The CA policy works within constraints determined by

an Adaptive Admission Control (AAC) algorithm. AAC algorithms monitor network conditions and dynamically adjust CA policy to optimize performance over various operating modes (e.g. normal or high loads, failure conditions). Specific cell-site architecture and CA algorithms used here are described in [1].

3 Adaptive Admission Control

For evaluation, metrics of interest are the forced termination rate (ftr) and the new call blocking rate (nbr). ftr is calculated as the ratio of the number of handover-call requests that were rejected to the total number of handover-call requests over a stated timeframe. nbr is the ratio of the number of new-call requests that were rejected to the total number of new-call requests over a stated timeframe. In general, it is desirable to have low values for ftr and nbr. However, since disconnection of an ongoing call is considered to cause greater user dissatisfaction than blocking a new-call request, lowering ftr is of higher priority. Thus, our performance objective is twofold, (1) minimize $ftr + nbr$, and (2) maintain an ftr/nbr ratio of 0.5.

We have been experimenting with two distributed AAC protocols that use a local guard band, denoted δ, to manage a tradeoff between ftr and nbr. δ specifies the percentage of channels within the cell that must be reserved for handover-call requests.

AAC 1:
if $ftr_{rt} > \gamma_{high}$ $\quad \delta+ = \Delta$
if $ftr_{rt} < \gamma_{low}$ $\quad \delta- = \Delta$

AAC 2:
if $ftr'_{rt} > \chi_{high}$ $\quad \delta+ = \Delta$
if $ftr'_{rt} < \chi_{low}$ $\quad \delta- = \Delta$

where the subscripts rt refers to real-time measurement of ftr and nbr. Both ftr_{rt} and nbr_{rt} are sliding window metrics. ftr'_{rt} and nbr'_{rt} are the derivatives of ftr and nbr respectively. Thus, AAC 1 compares the value ftr_{rt} to upper and lower thresholds, whereas AAC 2 compares the rate-of-change of ftr_{rt} to the threshold values. In both cases, increasing ftr_{rt} indicates increasing network congestion which calls for an increase in the guardband, while decreasing ftr_{rt} implies the opposite.

4 Visualization

Visualization tools have been constructed using the Visualization Toolkit (VTK) [3]. Our system has been designed using a spreadsheet style format. Each row represents a simulation run and within each row, an arbitrary number of metrics can be visualized (see Figure 3). This permits comparisons between a number of simulation runs, each run possibly using a different AAC algorithm. Two different types of visualizations have been used, (1) height fields, where the height corresponds to a metric value at a specific cell, (2) color maps, where the metric values are mapped into a set of colors (we use a rainbow color map, from blue to red). The application permits an arbitrary number of simulation datasets to be input and animated over time. VCR style controls permit the simulation run to be reviewed in a convenient manner.

5 Results

The cell site network used in our experiments consists of 105 cells arranged in a square grid. Using a nominal system-wide offered load of 50 calls/sec a number of experiments were performed towards determining the lowest values of ftr and nbr, calculated over one simulated hour, resulting in a 60 sec. sampling rate, 180 sec. sliding window size, $\gamma(AAC1) \in (0.02, 0.5)$, $\chi(AAC2) \in (0.0, 0.001)$. To compare AAC 1 and 2, we simulated the performance of each algorithm under different failure conditions, for a lightly loaded (40 calls/sec) and a heavily loaded (60 calls/sec) system. The simulation provides results for ftr and nbr, as calculated system-wide over one simulated hour (after 30 minute rampup). Each set of sampled data consists of 105 values for each of ftr_{rt}, δ, and nbr_{rt}, per sampled time step.

In general, we found the relative performance of the algorithms to be similar during different failures. AAC 1 consistently performed better during light loading and AAC 2 performed better during heavy loading. *What was not clear was "Why?"*. The following example[1] illustrates how application of data visualization proved essential to answering this question.

We simulated failure (loss of 75% of capacity) in 3 adjacent cells over a duration of 30 minutes for light, nominal and heavy loads. Figures 1-3 show snapshots of ftr, δ, and nbr for AAC 1 (top rows) and AAC 2 (bottom rows) taken at the specified time steps. Ideally, the system should operate as follows. With a significant peak in ftr (high network congestion), δ (guard band) should rise in the corresponding cell, followed by a similar peak in the nbr plane (meaning new calls are blocked during congestion). When the ftr diminishes, the δ peak should immediately diminish so as not to cause nbr to increase unnecessarily.

The visualizations illustrate that *AAC 1 is too sen-*

[1] Additional examples and animations may be found at http://www.cs.uncc.edu/~krs/mobvis

sitive to bursty traffic, while AAC 2 is not sensitive enough to recurring congestion during long failure periods. Figure 1 shows a nominal load run, shortly after the rampup period. The raised fields in the AAC 1 δ plane indicate cells for which the guardband has been increased due to bursts during the rampup period. In fact, note that the values of δ are not fully decreased to zero until time 2460. This is not justified, because since no significant peaks arise in the ftr plane during this time.

Figure 2 illustrates a light load run. At time 3000 sec., the failure has occurred (high peaks in ftr), and each of the algorithms react by increasing δ in 2-3 cells. At time 3060 (not shown), AAC 1 has fully responded to the failure (δ raised in 3 cells), but AAC 2 has already decreased δ in two cells. This implies that the failure condition is thought to be diminishing, when in fact, it is still in effect. AAC 1 tends to raise δ and keep it high throughout the 30 minute failure period. AAC 2 raises δ, then lowers it quickly. As a result, ftr will tend to repeatedly peak up and down throughout the failure period. AAC 2 tends not to respond after the initial failure impact. For example, time 3360 shows a significant ftr peak ignored by AAC 2. For lightly loaded cases, this causes AAC 2 to perform worse than AAC 1, since ftr peaks more frequently. However, for heavier loads, the slow response time of AAC 1 to diminish δ has a greater negative impact on performance, since a greater number of new calls are unnecessarily turned away. These results have helped us propose AAC 3.

AAC 3:
if $(ftr_{rt} > \gamma_{high})$ **OR** $(ftr'_{rt} > \chi_{high})$
$\delta+ = \Delta$
if $(ftr_{rt} < \gamma_{low})$ **OR**
$(ftr'_{rt} < 0$ **AND** $ftr_{rt} < \gamma_{high})$
$\delta- = \Delta$

The overall objective is to increase δ when either a fixed threshold is exceeded or a significant increase in value is detected for ftr_{rt}. A decrease in δ is made when either a fixed lower threshold is reached, or when the value of ftr_{rt} is decreasing and is also below a critical upper threshold value.

Snapshots comparing the performance of all three algorithms at heavy loads are illustrated in Figure 3. While AAC 3 is not yet optimal, it does seem to be a good compromise between AAC 1 and 2. At time 3000 sec, all algorithms respond appropriately. At time 3120, AAC 1 and 3 appropriately maintain higher δ values in the failed cells, while AAC 2 decreases δ too soon. At time 3360, AAC 1 continues to have increased values for δ even though the corresponding ftr values are flat. AAC 2 has exhibited a cycle of increased ftr, increased δ, decreased δ, decreased ftr, increased ftr, then no response from δ. AAC 3 has been gradually decreasing the raised δ values. At time 3480 AAC 2 has become completely unresponsive to ftr peaks. AAC 1 maintains unnecessarily high values for δ. AAC 3 has finally diminished δ appropriately.

6 Conclusions

Without the aid of the visualizations it is extremely difficult to fully understand algorithm behavior on a spatial (across all 105 cells) and temporal basis. Understanding algorithm behavior is crucial for the development of robust and stable adaptive algorithms. Continuing work includes developing more detailed visualizations to capture the interaction of multiple cooperating adaptive resource management algorithms.

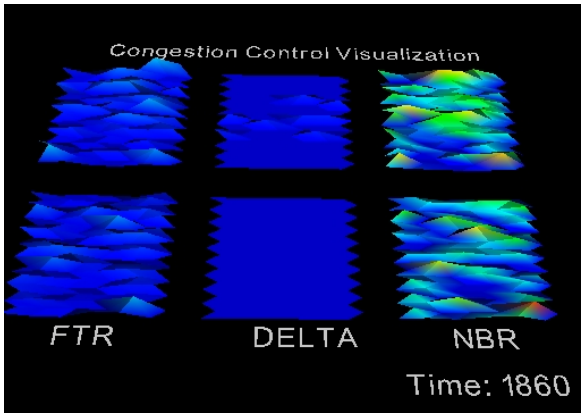
References

[1] T.A. Dahlberg and J. Jung. Survivable load sharing protocols: A simulation study. *ACM/Baltzer WINET. To appear*, 2000.

[2] E.E. Koutsofios, S.C. North, T. Truscott, and D.A. Keim. Visualizing large-scale telecommunications networks and services. In *Proceedings IEEE Visualization 1999, IEEE Computer Society*, 1999.

[3] W. Schroeder, K. Martin, and B. Lorensen. *The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics*. Prentice Hall, 2nd edition, 1998.

[4] V.Anupam, S.Dar, T.Leibfried, and E.Petajan. Dataspace: 3d visualization of large databases. In *Proceedings IEEE Information Visualization, IEEE Computer Society*, October 1995.



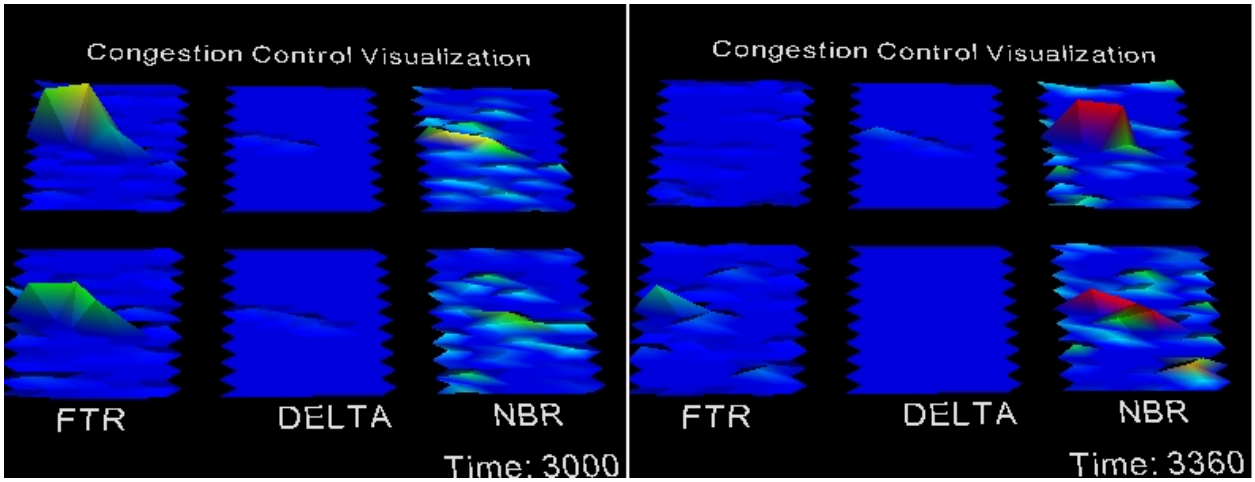


Figure 2: Light Load Performance

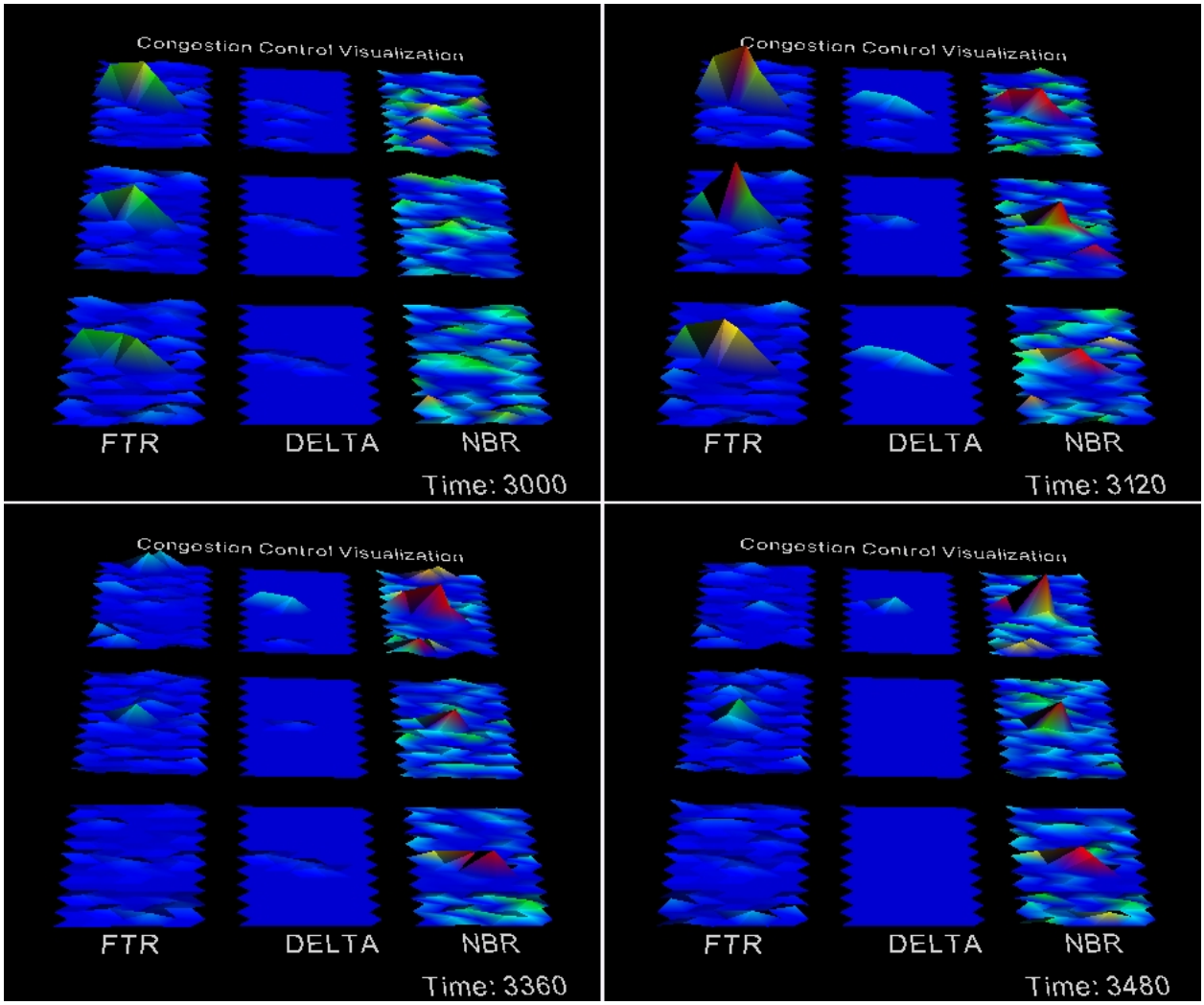


Figure 3: Heavy Load Performance